UECM1703 Introduction to Scientific Computing Marking Guide

TOPIC 5 Practical

UNIVERSITI TUNKU ABDUL RAHMAN

Faculty:	FES	Unit Code:	UECM1703
Course:	AM &FM	Unit Title:	Introduction Tt Scientific Computing
Year:	1&2	Lecturer:	Dr Yong Chin Khian
Session:	Oct 2022		

Q1. Consider the following matrices:

$$\mathbf{A} = \begin{bmatrix} 28.1 & 16.9 & 17.4 \\ 25.8 & 29.4 & 8.6 \\ 76.8 & 56.5 & 51.4 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 118.4 & 123.6 \\ 146.9 & 125.3 \end{bmatrix}.$$

Use python to compute matrix $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$. Then obtain the $(2,2)$ element of $\mathbf{C}$.

Ans.

$$\begin{aligned} &\mathbf{C} \\ &= \mathbf{A} \otimes \mathbf{B} \\ &= \begin{bmatrix} 28.1 & 16.9 & 17.4 \\ 25.8 & 29.4 & 8.6 \\ 76.8 & 56.5 & 51.4 \end{bmatrix} \otimes \begin{bmatrix} 118.4 & 123.6 \\ 146.9 & 125.3 \end{bmatrix} \\ &= \begin{bmatrix} 3327.0 & 3473.2 & 2001.0 & 2088.8 & 2060.2 & 2150.6 \\ 4127.9 & 3520.9 & 2482.6 & 2117.6 & 2556.1 & 2180.2 \\ 3054.7 & 3188.9 & 3481.0 & 3633.8 & 1018.2 & 1063.0 \\ 3790.0 & 3232.7 & 4318.9 & 3683.8 & 1263.3 & 1077.6 \\ 9093.1 & 9492.5 & 6689.6 & 6983.4 & 6085.8 & 6353.0 \\ 11281.9 & 9623.0 & 8299.8 & 7079.4 & 7550.7 & 6440.4 \end{bmatrix} \end{aligned}$$

$(2,2)$ element of $\mathbf{C} = 3520.9$.

```
import numpy as np
A = np.array([[28.1,16.9,17.4],[25.8,29.4,8.6],[76.8,56.5,51.4]])
B = np.array([[118.4,123.6],[146.9,125.3]])
C = np.kron(A,B)
print("C=",C)
ije = C[2-1,2-1]
print("(2,2) element of C = ", ije)
Output:
C= [[3327.0,3473.2,2001.0,2088.8,2060.2]
 [4127.9,3520.9,2482.6,2117.6,2556.1]
 [3054.7,3188.9,3481.0,3633.8,1018.2]
 [3790.0,3232.7,4318.9,3683.8,1263.3]
```

```
 [9093.1,9492.5,6689.6,6983.4,6085.8]]
(2,2) element of C =  3520.9
```

Q2. Consider the following matrices:

$$\mathbf{A} = \begin{bmatrix} 28.1 & 25.8 & 76.8 & 61.0 & 88.8 \\ 16.9 & 29.4 & 56.5 & 70.0 & 73.4 \\ 17.4 & 8.6 & 51.4 & 40.1 & 128.6 \\ 14.6 & 46.4 & 22.0 & 73.9 & 51.6 \\ 24.3 & 12.1 & 45.8 & 51.6 & 82.2 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 118.4 & 123.6 & 121.1 & 115.5 & 107.6 \\ 146.9 & 125.3 & 130.3 & 134.1 & 122.1 \\ 151.4 & 166.4 & 120.8 & 129.2 & 115.6 \\ 160.0 & 154.2 & 137.7 & 159.7 & 152.4 \\ 206.6 & 194.3 & 149.1 & 179.8 & 192.4 \end{bmatrix}.$$

Use python to compute matrix $\mathbf{C} = \mathbf{A}^{-1}\mathbf{B}$. Then obtain the determinat of $\mathbf{C}$.

Ans.

$$\mathbf{C} = \mathbf{A}^{-1}\mathbf{B}$$

$$= \begin{bmatrix} 0.0424 & 0.0522 & 0.0178 & -0.0437 & -0.0027 \\ -0.0883 & -0.0449 & 0.0178 & 0.0474 & -0.0069 \\ -0.0321 & 0.0027 & 0.0029 & -0.0104 & 0.0140 \\ 0.0218 & 0.0339 & -0.0149 & -0.0123 & 0.0046 \\ 0.0635 & -0.0514 & -0.0261 & 0.0324 & -0.0048 \end{bmatrix} \begin{bmatrix} 118.4 & 123.6 & 121.1 & 115.5 & 107.6 \\ 146.9 & 125.3 & 130.3 & 134.1 & 122.1 \\ 151.4 & 166.4 & 120.8 & 129.2 & 115.6 \\ 160.0 & 154.2 & 137.7 & 159.7 & 152.4 \\ 206.6 & 194.3 & 149.1 & 179.8 & 192.4 \end{bmatrix}$$

$$= \begin{bmatrix} 7.8332 & 7.4809 & 7.6673 & 6.7329 & 5.8137 \\ -8.2143 & -7.6266 & -8.9099 & -7.6066 & -7.0448 \\ -1.7402 & -2.0333 & -2.5351 & -2.1194 & -1.6845 \\ 4.286 & 3.4585 & 4.2493 & 4.0013 & 3.772 \\ 0.198 & 1.1177 & 1.5761 & 1.3709 & 1.5445 \end{bmatrix}$$

$$\begin{vmatrix} 7.8332 & 7.4809 & 7.6673 & 6.7329 & 5.8137 \\ -8.2143 & -7.6266 & -8.9099 & -7.6066 & -7.0448 \\ -1.7402 & -2.0333 & -2.5351 & -2.1194 & -1.6845 \\ 4.286 & 3.4585 & 4.2493 & 4.0013 & 3.772 \\ 0.198 & 1.1177 & 1.5761 & 1.3709 & 1.5445 \end{vmatrix} = -1.5879228935531118$$

```
import numpy as np
A = np.array([[28.1,16.9, 17.4, 14.6, 24.3],
             [25.8,29.4, 8.6, 46.4, 12.1],
             [76.8,56.5, 51.4, 22.0, 45.8],
             [61.0,70.0, 40.1, 73.9, 51.6],
             [88.8,73.4, 128.6, 75.0, 82.2]])
B = np.array([[118.4,123.6, 121.1, 115.5, 107.6],
             [146.9,125.3, 130.3, 134.1, 122.1],
             [151.4,166.4, 120.8, 129.2, 115.6],
```

```
                [160.0,154.2, 137.7, 159.7, 152.4],
                [206.6,194.3, 149.1, 179.8, 192.4]])
C = np.linalg.inv(A)@B
print("C=",C)
DetC = np.linalg.det(C)
print("(|C|=", DetC)
Output:
C= [[7.8332,7.4809,7.6673,6.7329,5.8137]
 [-8.2143,-7.6266,-8.9099,-7.6066,-7.0448]
 [-1.7402,-2.0333,-2.5351,-2.1194,-1.6845]
 [4.286,3.4585,4.2493,4.0013,3.772]
 [0.198,1.1177,1.5761,1.3709,1.5445]]
|C|= -1.5879228935531118
```

Q3. Consider the following matrix:

$$\mathbf{C} = \begin{bmatrix} 32.32 & -33.71 & 81.45 & -20.36 & -2.66 \\ -33.71 & 227.97 & -129.15 & 193.79 & -232.12 \\ 81.45 & -129.15 & 390.71 & -72.07 & 91.93 \\ -20.36 & 193.79 & -72.07 & 189.62 & -269.92 \\ -2.66 & -232.12 & 91.93 & -269.92 & 513.0 \end{bmatrix}.$$

Use python to obtain the eigen value and eigen vector of $\mathbf{C}$. Then find the largest values of the eigen values.

Ans.

Eigen values of $\mathbf{C}$ is $\begin{bmatrix} 875.0425 & 367.2123 & 98.4886 & -0.0008 & 12.8773 \end{bmatrix}$

Eigen vector of $\mathbf{C}$

$$\begin{bmatrix} -0.0583 & -0.2155 & -0.1601 & 0.4074 & 0.8710 \\ 0.4517 & 0.0500 & 0.6888 & -0.4275 & 0.3692 \\ -0.3287 & -0.8875 & 0.2609 & -0.1411 & -0.1276 \\ 0.4399 & -0.1380 & 0.3512 & 0.7595 & -0.2954 \\ -0.7007 & 0.3799 & 0.5555 & 0.2336 & 0.0400 \end{bmatrix}$$

Largest eigen value = 875.0425

```
import numpy as np
C = np.array([[32.32,-33.71, 81.45, -20.36, -2.66],
[-33.71,227.97, -129.15, 193.79, -232.12],
[81.45,-129.15, 390.71, -72.07, 91.93],
[-20.36,193.79, -72.07,189.62, -269.92],
[-2.66,-232.12, 91.93, -269.92, 513.0]])
w,v = np.linalg.eig(C)
```

```
print("Eigen values of C =",w)
print("Eigen vector of C =",v)
LV = np.max(w)
print("Largest eigen vector = ",LV)
Output:
Eigen values of C = [ 875.0425,367.2123,98.4886,-0.0008,12.8773]
Eigen vector of C = [[-0.0583,-0.2155,-0.1601,0.4074,0.8710],
[0.4517,0.0500,0.6888,-0.4275,0.3692],
[-0.3287,-0.8875,0.2609,-0.1411,-0.1276],
[0.4399,-0.1380,0.3512,0.7595,-0.2954],
[-0.7007,0.3799,0.5555,0.2336,0.0400]]
Largest eigen vector =  875.0425
```

Q4. Consider the following linear system:

$$\begin{aligned} 33.1w + 21.9x + 22.4y + 19.6z &= 94.8 \\ 30.8w + 34.4x + 13.6y + 51.4z &= 79.4 \\ 66.8w + 46.5x + 41.4y + 12.0z &= 134.6 \\ 70.0w + 79.0x + 49.1y + 82.9z &= 81.0 \end{aligned}$$

(a) Write the above system in the form $\mathbf{AX} = \mathbf{b}$.

Ans.

$$\begin{bmatrix} 33.1 & 21.9 & 22.4 & 19.6 \\ 30.8 & 34.4 & 13.6 & 51.4 \\ 66.8 & 46.5 & 41.4 & 12.0 \\ 70.0 & 79.0 & 49.1 & 82.9 \end{bmatrix} \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 94.8 \\ 79.4 \\ 134.6 \\ 81.0 \end{bmatrix}$$

(b) Obtain the solution to the system above using matrix inversion.

Ans.

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

$$= \begin{bmatrix} 0.0228 & 0.0676 & 0.0260 & -0.0511 \\ -0.1318 & -0.0111 & 0.0360 & 0.0328 \\ 0.0970 & -0.1003 & -0.0506 & 0.0466 \\ 0.0488 & 0.0129 & -0.0263 & -0.0037 \end{bmatrix} \begin{bmatrix} 94.8 \\ 79.4 \\ 134.6 \\ 81.0 \end{bmatrix}$$

$$= \begin{bmatrix} 6.8949 \\ -5.8657 \\ -1.8127 \\ 1.8185 \end{bmatrix}$$

```
import numpy as np
```

```
A = np.array([[33.1,21.9,22.4,19.6],[30.8,34.4,13.6,51.4],
[66.8,46.5,41.4,12.0],[70.0,79.0,49.1,82.9]])
b = np.array([[94.8],[79.4],[134.6],[81.0]])
x = np.linalg.inv(A)@b
print(x)
Output:
[[6.8949]
 [-5.8657]
 [-1.8127]
 [1.8185]]
```

(c) Compute $93.8w + 114.6x + 104.4y + 82.0z$.

Ans.

$93.8w + 114.6x + 104.4y + 82.0z = 93.8(6.8949) + 114.6(-5.8657) + 104.4(-1.8127) + 82.0(1.8185) = -65.5965$

```
import numpy as np
A = np.array([[33.1,21.9,22.4,19.6],[30.8,34.4,13.6,51.4],
[66.8,46.5,41.4,12.0],[70.0,79.0,49.1,82.9]])
b = np.array([[94.8],[79.4],[134.6],[81.0]])
x = np.linalg.inv(A)@b
anew = np.array([[93.8,114.6,104.4,82.0]])
xnew = np.inner(x, anew)
print("xnew = ", xnew)
Output:
xnew = -65.5965
```

Q5. You fit the following data to $Y = \beta_0 + \beta_1 X + \epsilon$.

x	y
71	36
34	25
35	25
26	23
58	32
6	17

(a) Write the model above in the form $\mathbf{y} = \mathbf{x}\boldsymbol{\beta} + \boldsymbol{\epsilon}$

Ans.

$$\begin{bmatrix}36\\25\\25\\23\\32\\17\end{bmatrix} = \begin{bmatrix}1 & 71\\1 & 34\\1 & 35\\1 & 26\\1 & 58\\1 & 6\end{bmatrix} \begin{bmatrix}\beta_0\\\beta_1\end{bmatrix} \begin{bmatrix}+\epsilon_1\\\epsilon_2\\\epsilon_3\\\epsilon_4\\\epsilon_5\\\epsilon_6\end{bmatrix}$$

(b) Write the Python commands and output using matrix formulation to obtain the estimate of β_0 and β_1.

Ans.

```
import numpy as np
y = np.array([36,25,25,23,32,17])
x = np.array([[1,71],[1,34],[1,35],[1,26],[1,58],[1,6]])
xtx = x.T@x
xty = x.T@y
b = np.linalg.inv(xtx)@xty
print(b)
Output:
[15.17739930382892 0.29102436598707104]
```

(c) You are given that $R^2 = \frac{SSR}{SST}$ and adjusted R^2, $R^2_{Adj} = 1 - \frac{SSE/(n-p)}{SST/(n-1)}$, where

- n is the number of observations.
- p is the number of parameters in the model.
- $SSR = \boldsymbol{b}^T\mathbf{x}^\mathbf{T}\mathbf{y} - \frac{1}{n}\mathbf{y}^\mathbf{T}\mathbf{J}\mathbf{y}$, where $\mathbf{J}$ is an $n \times n$ matrix of one.
- $SSE = \mathbf{y}^\mathbf{T}\mathbf{y} - \boldsymbol{b}^T\mathbf{x}^\mathbf{T}\mathbf{y}$.
- $SST = \mathbf{y}^\mathbf{T}\mathbf{y} - \frac{1}{n}\mathbf{y}^\mathbf{T}\mathbf{J}\mathbf{y}$.

Write the Python commands and output to calculated R^2 and adjusted R^2.

Ans.

```
import numpy as np
y = np.array([36,25,25,23,32,17])
x = np.array([[1,71],[1,34],[1,35],[1,26],[1,58],[1,6]])
n = len(y)
p = 2
xtx = x.T@x
xty = x.T@y
b = np.linalg.inv(xtx)@xty
```

```
print("beta hat =", b)
yty = y.T@y
bxty = b.T@xty
J = np.ones((n,n))
ytJy = y.T@J@y
SSR = bxty-ytJy/n
SSE = yty-bxty
SST = yty-ytJy/n
print("SSE=", SSE)
print("SSR=", SSR)
print("SST=",SST)
RSq = SSR/SST
AdjRSq = 1-(SSE/(n-p))/(SST/(n-1))
print("Rsq=",RSq)
print("AdjRsq=", AdjRSq)

Output:
beta hat = [15.17739930382892 0.29102436598707104]
SSE= 0.23731974142538093
SSR= 227.09601359190765
SST= 227.33333333333303
RSq= 0.9989560715186567
AdjRSq= 0.9986950893983209
```

Q6. Consider the data below:

y	x_1	x_2	x_3	x_4
275.4	7.9	46.2	15.4	30.2
181.3	4.6	23.9	8.7	16.8
184.7	4.7	24.7	8.9	17.3
162.2	3.9	19.3	7.3	14.1
243.8	6.8	38.7	13.1	25.7
110.2	2.0	7.0	3.6	6.7
280.9	8.1	47.4	15.7	31.0
175.0	4.4	22.4	8.2	15.9
220.7	6.0	33.2	11.5	22.4
196.7	5.1	27.5	9.8	19.0

Determine the predicted value for the mean score of y with $x_1 = 5.9$, $X_2 = 30.8$, $x_3 = 10.2$, and $x_4 = 19.0$.

Ans.

$\hat{Y} = 64.1535 + 1.2658(5.9) + 0.4089(30.8) + 0.9010(10.2) + 5.5795(19.0) = 199.4146$

```
import numpy as np
import statsmodels.api as sm
y = np.array([275.4, 181.3, 184.7, 162.2, 243.8,
110.2, 280.9, 175.0, 220.7,196.7])
n = len(y)
x0 = np.ones(n).reshape(-1,1)
x1 = np.array([7.9, 4.6, 4.7, 3.9, 6.8,
2.0, 8.1, 4.4, 6.0, 5.1]).reshape(-1,1)
x2 = np.array([46.2, 23.9, 24.7, 19.3, 38.7,
7.0, 47.4, 22.4, 33.2, 27.5]).reshape(-1,1)
x3 = np.array([15.4, 8.7, 8.9, 7.3, 13.1,
3.6, 15.7, 8.2, 11.5, 9.8]).reshape(-1,1)
x4 = np.array([30.2, 16.8, 17.3, 14.1, 25.7,
6.7, 31.0, 15.9, 22.4, 19.0]).reshape(-1,1)
x = np.hstack((x0, x1,x2,x3,x4))
xtx = x.T@x
xty = x.T@y
coef = np.linalg.inv(xtx)@xty
xh = np.array([1,5.9, 30.8, 10.2, 19.0])
yh = np.inner(xh, coef)
print("beta hat=", coef)
print("y hat=", yh)
Output:
beta hat= [64.1535,1.2658,0.4089,0.9010,5.5795]
y hat= 199.4146
```

Q7. Consider the data shown below:

y	x	y	x
19	12	19	13
19	13	18	9
18	9	19	14
19	14	19	12
18	11	19	13
17	8	19	13
18	10	18	9
19	14	19	13

You fit the above data to $y = \beta_0 + \beta_1 x + \epsilon$. You are given that:

- n is the number of observations.
- p is the number of parameters in the model.
- $\boldsymbol{b} = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$.
- $SSE = \mathbf{y^T y} - \boldsymbol{b}^T \mathbf{X^T y}$.
- $MSE = \frac{SSE}{n-p}$.

Compute MSE.

Ans.

$MSE = 0.06685432793528889$

```
import numpy as np
y = np.array([19,19,18,19,18,17,18,19,
19,18,19,19,19,19,18,19])
x = np.array([[1,12],[1,13],[1,9],[1,14],[1,11],[1,8],
[1,10],[1,14],[1,13],[1,9],[1,14],[1,12],[1,13],
[1,13],[1,9],[1,13]])
xtx = x.T@x
xty = x.T@)
b = np.linalg.inv(xtx)@xty
n = len(y)
p = 2
xtx = x.T@x
xty = x.T@y
b = np.linalg.inv(xtx)@xty
print("beta hat =", b)
yty = y.T@y
bxty = b.T@xty
J = np.ones((n,n))
ytJy = y.T@J@y
SSE = yty-bxty
MSE = SSE/(n-2)
print("MSE=", MSE)
Output:
MSE = 0.06685432793528889
```

Q8. Consider the data shown below:

y	x	z	y	x	z
28	17	31	25	12	25
26	13	27	25	11	24
25	12	26	23	8	22
23	7	20	26	14	28
25	11	25	26	14	28
25	11	24	28	17	31
25	12	25	25	12	26
24	10	24	25	11	24

You fit the above data to $y = \beta_0 + \beta_1 x + \beta_2 z + \epsilon$. You are given that:

- n is the number of observations.
- p is the number of parameters in the model.
- $\boldsymbol{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$.
- $SSR = \boldsymbol{b}^T \mathbf{x}^\mathbf{T} \mathbf{y} - \frac{1}{n} \mathbf{y}^\mathbf{T} \mathbf{J} \mathbf{y}$, where $\mathbf{J}$ is an $n \times n$ matrix of one.

Compute SSR.

Ans.

SSR= 28.113636365085767

```
import numpy as np
y = np.array([28,26,25,23,25,25,
25, 24,25,25,23,26,26,
28,25,25])
x = np.array([[1,17,31],[1,13,27],[1,12,
26],[1,7,20],[1,11,25],[1,11,
24],[1,12,25],[1,10,24],[1,12,
25],[1,11,24],[1,8,22],[1,14,
28],[1,14,28],[1,17,31],[1,12,
26],[1,11,24]])
xtx = x.T@x
xty = x.T@y
b = np.linalg.inv(xtx)@xty
n = len(y)
p = 3
xtx = x.T@x
xty = x.T@y
b = np.linalg.inv(xtx)@xty
print("beta hat =", b)
yty = y.T@y
bxty = b.T@xty
J = np.ones((n,n))
ytJy = y.T@J@y
SSR = bxty-ytJy/n
print("SSR=", SSR)

Output:
SSR = 28.113636365085767
```

Q9. The world 10 largest companies yield the following data (in billion):

Company	x_1 = sales	x_2 = profits	x_3 = assets
1	244.4	25.8	1135.8
2	128.1	13.0	554.4
3	132.4	13.5	576.2
4	104.1	10.4	434.5
5	204.9	21.4	938.5
6	40.2	3.4	115.1
7	250.6	26.4	1167.2
8	120.3	12.2	515.6
9	176.5	18.3	796.5
10	146.9	15.1	648.7

Derive the sample covariance matrix using the NumPy package, then provide $cov(x_2, x_2)$.

Ans.

$\mathbf{S_n}$ = [[4.26196044e+03 4.66340000e+02 2.13079044e+04] [4.66340000e+02 5.10272222e+01 2.33149167e+03] [2.13079044e+04 2.33149167e+03 1.06530052e+05]]

$cov(x_2, x_2) = \boxed{51.027222222222214}$

```
import numpy as np
x1 = [244.4,128.1,132.4,104.1,204.9,40.2,250.6,120.3,176.5,146.9]
x2 = [25.8,13.0,13.5,10.4,21.4,3.4,26.4,12.2,18.3,15.1]
x3 = [1135.8,554.4,576.2,434.5,938.5,115.1,1167.2,515.6,796.5,648.7]
data = np.array([x1, x2, x3])
cov_matrix = np.cov(data, bias=False)
print(cov_matrix)
covij = cov_matrix[1, 1]
print("cov(x_2, x_2)=",covij)
Output:
cov(x_2, x_2)= 51.027222222222214
```

Q10. Consider the data shown below:

y	x	z	y	x	z
27	16	30	26	13	26
24	10	24	25	11	25
24	11	24	27	16	30
23	9	22	22	6	19
26	14	28	25	11	25
22	6	19	24	11	24
28	17	31	27	15	29
24	10	23	23	9	22

You fit the above data to $y = \beta_0 + \beta_1 x + \beta_2 z + \epsilon$. You are given that:

- n is the number of observations.
- p is the number of parameters in the model.
- $\boldsymbol{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$.
- $SSE = \mathbf{y^T y} - \boldsymbol{b}^T \mathbf{x^T y}$.
- $MSE = \frac{SSE}{n-p}$
- $SE(b_j) = \sqrt{MSE \times C_{jj}}$, where C_{jj} is the diagonal element of the $(\mathbf{X^T X})^{-1}$ corresponding to b_j.

Compute $SE(b_1)$.

Ans.

$SE(\hat{\beta}_1) = 0.25139635250620596$

```
from scipy import sqrt,linalg
import numpy as np
y = np.array([27,24,24,23,26,22,
28,24,26,25,27,22,25,
24,27,23])
x = np.array([[1,16,30],[1,10,24],[1,11,
24],[1,9,22],[1,14,28],[1,6,
19],[1,17,31],[1,10,23],[1,13,
26],[1,11,25],[1,16,30],[1,6,
19],[1,11,25],[1,11,24],[1,15,
29],[1,9,22]])
xtx = x.T@x
xty = x.T@y
b = linalg.inv(xtx)@xty
n = len(y)
```

```
p = 3
print("beta hat =", b)
yty = y.T@y
bxty = b.T@xty
SSE = yty-bxty
MSE = SSE/(n-p)
Cjj = linalg.inv(xtx)[1,1]
SEbeta = sqrt(MSE*Cjj)
print("SEbeta=", SEbeta)
Output:
SEbeta= 0.25139635250620596
```

Q11. A researcher believes that the number of days the ozone levels exceeded 0.2ppm (y) depends on the seasonal meteorological index (x). The following table gives the data.

Index	16.3	16.8	18.2	17.4	17.0	17.4	14.1	16.8	17.6	17.1
Days	86	114	113	116	79	82	67	80	72	58

You fit the above data to $y = \beta_0 + \beta_1 x + \epsilon$, where y is the number of days the ozone levels exceeded 0.2ppm, and x is the seasonal meteorological index.

- n is the number of observations.
- p is the number of parameters in the model.
- $SSR = \boldsymbol{b}^T\mathbf{x}^\mathbf{T}\mathbf{y} - \frac{1}{n}\mathbf{y}^\mathbf{T}\mathbf{J}\mathbf{y}$, where $\mathbf{J}$ is an $n \times n$ matrix of one.
- $SSE = \mathbf{y}^\mathbf{T}\mathbf{y} - \boldsymbol{b}^T\mathbf{x}^\mathbf{T}\mathbf{y}$.
- $MSE = \frac{SSE}{n-p}$.
- $MSR = \frac{SSR}{p-1}$.

Use Python to calculated F.

Ans.

$F = 1.5736127799672888$

```
import numpy as np
y = np.array([86,114,113,116,79,82,67,80,72,58])
x = np.array([[1,16.3],[1,16.8],[1,18.2],[1,17.4],[1,17.0],
             [1,17.4],[1,14.1],[1,16.8],[1,17.6],[1,17.1]])
n = len(y)
p = 2
xtx = x.T@x
```

```
xty = x.T@y
b = np.linalg.inv(xtx)@xty
print("beta hat =", b)
yty = y.T@y
bxty = b.T@xty
J = np.ones((n,n))
ytJy = y.T@J@y
SSR = bxty-ytJy/n
SSE = yty-bxty
MSR = SSR/(p-1)
MSE = SSE/(n-p)
F = MSR/MSE
print("F=", F)
Output:1.5736127799672888
```

Q12. You are given the following data:

No.	x_1	x_2	x_3
1	13.7	1.3	60.7
2	15.6	1.5	70.0
3	4.8	0.3	16.0
4	24.4	2.4	114.1
5	6.6	0.5	24.9
6	11.5	1.0	49.6
7	18.5	1.8	84.6
8	9.4	0.8	39.1
9	10.4	0.9	44.0
10	11.2	1.0	47.9

Derive the sample covariance matrix($\mathbf{S_n}$) using the NumPy package, then determine the determinant of $\mathbf{S_n}$.

Ans.

$\mathbf{S_n}$ = [[3.32610000e+01 3.59722222e+00 1.66560111e+02] [3.59722222e+00 3.89444444e-01 1.80138889e+01] [1.66560111e+02 1.80138889e+01 8.34085444e+02]]

$|\mathbf{S_n}| = \boxed{9.697325103319101\text{e-}05}$

```
import numpy as np
```

```
x1 = [13.7,15.6,4.8,24.4,6.6,11.5,18.5,9.4,10.4,11.2]
x2 = [1.3,1.5,0.3,2.4,0.5,1.0,1.8,0.8,0.9,1.0]
x3 = [60.7,70.0,16.0,114.1,24.9,49.6,84.6,39.1,44.0,47.9]
data = np.array([x1, x2, x3])
cov_matrix = np.cov(data, bias=False)
DetSn = np.linalg.det(cov_matrix)
print("Det(S_n) = ",DetSn)
Output:
Det(S_n) =  9.697325103319101e-05
```